Operadores

APRESENTAÇÃO

Os operadores são símbolos utilizados para escrever expressões. Essas expressões podem ser de atribuição, aritméticas, lógicas e relacionais. Além de tipos diferentes, as expressões podem conter vários operadores, o que implica entender qual é a precedência entre esses operadores.

Nesta Unidade de Aprendizagem, você irá compreender como funcionam os operadores e quais são eles em C. Além disso, você irá estudar os conceitos de incremento, como ele pode ser feito e seu importante papel na arte da programação de computadores.

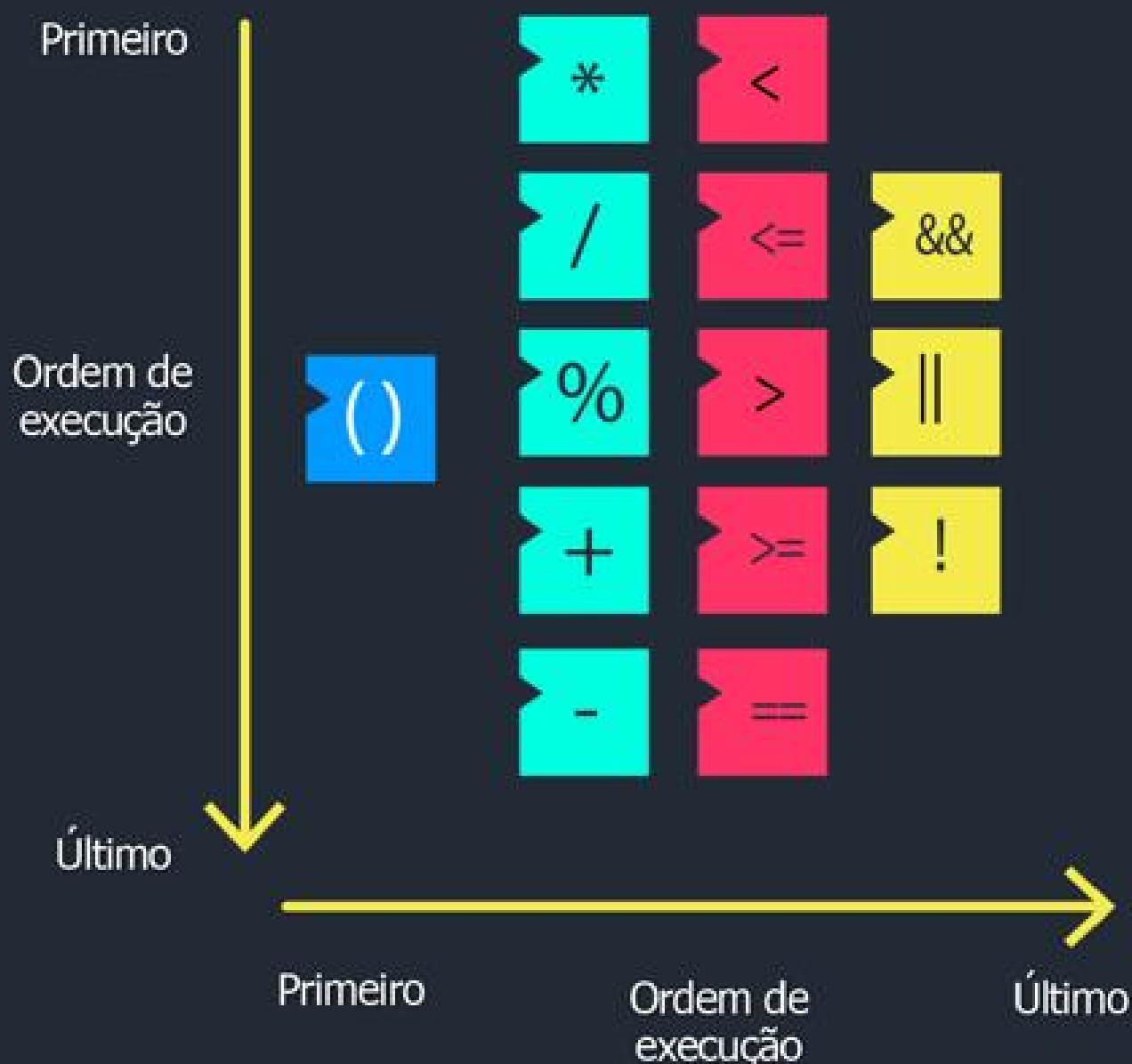
Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Compreender o uso dos operadores de atribuição, aritméticos, relacionais e lógicos.
- Construir expressões com vários operadores e precedência.
- Desenvolver expressões que utilizem operadores de pré e pós-incremento/decremento.

INFOGRÁFICO

Como resolver expressões com vários operadores de tipos diferentes? O infográfico, a seguir, apresenta dicas simples de como fazê-lo e ter a resposta correta para expressões com muitos operadores usando, para isso, a ordem de precedência.

COMO RESOLVER EXPRESSÕES COM VÁRIOS TIPOS DE OPERADORES: AVALIANDO A ORDEM DE PRECEDÊNCIA.

1. Os parênteses, quando existentes, são os primeiros a serem resolvidos.

2. Operações aritméticas precisam ser feitas primeiramente, seguindo sua ordem de precedência.

3. Em segundo lugar, as operações relacionais com sua ordem de precedência.

4. Por fim, as operações lógicas na ordem: Não, E e Ou.

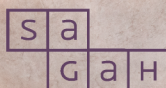
CONTEÚDO DO LIVRO

Os operadores são peças fundamentais para realizar diversas tarefas no mundo da programação utilizando expressões. Essas expressões podem ser de atribuição, relacionais ou lógicas, e podem configurar expressões de um tipo ou haver expressões com vários operadores de vários tipos.

Além disso, existe uma forma rápida de realizar uma tarefa bastante comum na computação: incremento e decremento de uma variável.

Leia mais no capítulo Operadores, da obra *Algoritmos de programação*, base teórica para esta Unidade de Aprendizagem.

Boa leitura.

ALGORITMOS DE PROGRAMAÇÃO

Marcela Santos

Operadores

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Reconhecer o uso dos operadores de atribuição, aritméticos, relacionais e lógicos.
- Construir expressões com vários operadores e precedência.
- Desenvolver expressões que utilizem operadores de pré e pós-incremento/decremento.

Introdução

Os operadores são símbolos que são utilizados para escrevermos expressões, as quais podem ser de atribuição, aritméticas, lógicas e relacionais. Além de tipos diferentes, as expressões podem conter vários operadores, o que implica entendermos qual a precedência entre estes.

Neste capítulo, você vai compreender como funcionam os operadores e quais são eles em C. Além disso, você será apresentado aos conceitos de incremento, como ele pode ser feito e seu papel importante na arte da programação de computadores.

Operadores de atribuição, aritméticos, relacionais e lógicos

Como foi visto anteriormente, uma variável é um espaço de memória de um determinado tipo, que vai guardar um valor seja do tipo inteiro, real ou caractere. Existe uma função predefinida em uma biblioteca padrão do C, que nos permite guardar um valor digitado pelo usuário em uma variável declarada, a função scanf.

Mas como podemos armazenar um valor em uma variável, sendo que ele não foi digitado pelo usuário — como, por exemplo, no momento que inicializamos uma variável? Observe a Figura 1.

```
1  #include <stdio.h>
2  int main(){
3      int idade=34;
4      int anoAtual=2018;
5      int anoNascimento;
6      anoNascimento=2018-idade;
7      printf("Oi,voce nasceu em %d\n",anoNascimento);
8      return 0;
9  }
```

Figura 1. Código para exemplificar o operador de atribuição.

As linhas 3 e 4 mostram a declaração e inicialização de duas variáveis; tomemos a linha 3 como exemplo. Essa linha pode ser traduzida como: "armazeno o valor 34 dentro da variável idade". Para realizar essa operação, utilizamos o operando = (igual), de atribuição. A sintaxe de uso deste operando em uma expressão de atribuição pode ser vista na Figura 2.

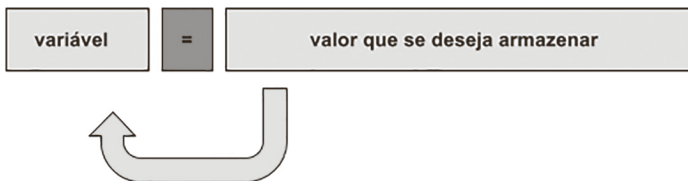



Figura 2. Sintaxe do operador de atribuição.

Esse valor armazenado pode ser um valor direto (como no exemplo), ou uma expressão aritmética, lógica ou relacional.

Todas as operações aritméticas, já conhecidas da nossa matemática clássica, podem ser usadas em programação. Cada linguagem de programação tem seus operandos para representar essas operações — em alguns casos, são os mesmos que já conhecemos em outros, existindo uma pequena diferença.

Para exemplificar, vamos desenvolver um programa em C que realiza as quatro operações básicas com dois números inteiros, conforme representado na Figura 3.

```
1  #include <stdio.h>
2  int main(){
3      int numero1,numero2;
4      int soma,subtracao,multiplicacao,divisao;
5
6      printf("*** Calculadora Modelo 1 ***\n");
7      printf("Digite o primeiro numero: ");
8      scanf("%d",&numero1);
9
10     printf("Digite o segundo numero: ");
11     scanf("%d",&numero2);
12
13     soma=numero1+numero2;
14     subtracao=numero1-numero2;
15     multiplicacao=numero1*numero2;
16     divisao=numero1/numero2;
17
18     printf("Numero 1: %d\n",numero1);
19     printf("Numero 2: %d\n",numero2);
20     printf("Soma: %d\n",soma);
21     printf("Subtracao entre numero 1 e numero 2: %d\n",subtracao);
22     printf("Multiplicacao: %d\n",multiplicacao);
23     printf("Divisao entre numero 1 e numero 2: %d\n",divisao);
24
25     return 0;
26 }
```

Figura 3. Código de uma calculadora com as quatro operações aritméticas.

Ao executar esse programa e digitando os valores 5 e 2 para as variáveis numero1 e numero2, respectivamente, teremos a seguinte resposta, na Figura 4:

```
*** Calculadora Modelo 1 ***
Digite o primeiro numero: 5
Digite o segundo numero: 2
Numero 1: 5
Numero 2: 2
Soma: 7
Subtracao entre numero 1 e numero 2: 3
Multiplicacao: 10
Divisao entre numero 1 e numero 2: 2
```

Figura 4. Saída da calculadora com as quatro operações aritméticas.

Um ponto interessante a se avaliar essa saída de dados é a resposta à operação de divisão. Em C, ao se dividir dois números inteiros, o resultado também é um número inteiro, mesmo que a resposta da divisão seja um número real, como no caso de 5 dividido por 2. Assim, a resposta aparece truncada, que quer dizer somente com a parte inteira.

Para resolvermos isso, podemos declarar as variáveis como do tipo float e delimitar o número de casas decimais depois da vírgula. Isso pode ser visto, a seguir, na Figura 5 e na Figura 6.

```
1  #include <stdio.h>
2  int main(){
3      float numero1,numero2;
4      float soma,subtracao,multiplicacao,divisao;
5
6      printf("*** Calculadora Modelo 1 ***\n");
7      printf("Digite o primeiro numero: ");
8      scanf("%f",&numero1);
9
10     printf("Digite o segundo numero: ");
11     scanf("%f",&numero2);
12
13     soma=numero1+numero2;
14     subtracao=numero1-numero2;
15     multiplicacao=numero1*numero2;
16     divisao=numero1/numero2;
17
18     printf("Numero 1: %.2f\n",numero1);
19     printf("Numero 2: %.2f\n",numero2);
20     printf("Soma: %f\n",soma);
21     printf("Subtracao entre numero 1 e numero 2: %.2f\n",subtracao);
22     printf("Multiplicacao: %.2f\n",multiplicacao);
23     printf("Divisao entre numero 1 e numero 2: %.2f\n",divisao);
24
25     return 0;
26 }
```

Figura 5. Código de uma calculadora com as quatro operações aritméticas usando o tipo float.

```
*** Calculadora Modelo 1 ***
Digite o primeiro numero: 5
Digite o segundo numero: 2
Numero 1: 5.00
Numero 2: 2.00
Soma: 7.000000
Subtracao entre numero 1 e numero 2: 3.00
Multiplicacao: 10.00
Divisao entre numero 1 e numero 2: 2.50
```

Figura 6. Saída da calculadora com as quatro operações aritméticas usando o tipo float.

Além disso, existe um operando que permite que se armazene o resto da divisão, também chamado de operação módulo. Esta operação é realizada com o operando % (por cento). Para exemplificarmos, vamos mostrar o resto da divisão de um número digitado pelo usuário, por 2, conforme Figura 7. Como sabemos, se o resto for 0, o número é par. Vamos ao código em C!

```
1  #include <stdio.h>
2  int main(){
3      int numero,resto;
4      printf("Digite o numero: ");
5      scanf("%d", &numero);
6      resto=numero%2;
7      printf("Resto divisao: %d\n",resto);
8      return 0;
9  }
```

Figura 7. Uso do operador módulo.

Como nós estamos utilizando a linguagem C, a seguir, você pode ver, na Tabela 1, a operação aritmética e o seu operando, bem como um exemplo de expressão de utilização em um programa de computador escrito na linguagem C.

Tabela 1. Operadores aritméticos.

Operação	Operador	Expressão
Soma	+	a= b+1;
Subtração	-	a= b-25 ;
Multiplicação	*	a= a*2 ;
Divisão	/	a=a/2 ;
Módulo	%	a= 4%2

Além dos operadores de atribuição e aritméticos, existem os operadores lógico-relacionais. Os operadores relacionais (Tabela 2, a seguir) são utilizados quando se precisa comparar variáveis e/ou valores do mesmo tipo. Uma operação relacional tem como resultado os valores lógicos verdadeiro ou falso.

Variáveis que podem ter somente dois tipos, como verdadeiro ou falso, são conhecidas em programação como variáveis do tipo booleano. Booleano vem da álgebra que usa esses valores, criada pelo matemático George Boole.

O detalhe é que, em C até o compilador de 1999, nós não tínhamos esse tipo, sendo necessário que os programadores simulassem. Atualmente, basta que seja adicionada a biblioteca stdbool.h. Vamos usar bastante os operadores relacionais e o tipo bool que pode ter o valor 1 ou 0, quando estivermos utilizando as estruturas de seleção. Por agora, vale a pena conhecer o operando e a sua função.

Link

Para saber mais sobre o tipo bool e a biblioteca stdbool, acesse:

https://goo.gl/5g1xr2

Tabela 2. Operadores relacionais.

Operador relacional	Exemplo : sejam a e b duas variáveis do mesmo tipo, como, por exemplo, a = 10 e b =5	Descrição	Resultado
==	a == b	Avalie se a é igual b	0
>	a > b	Avalie se a é maior que b	1
<	a < b	Avalie se a é menor que b	0
>=	a > = b	Avalie se a é maior ou menor que b	1
<=	a <= b	Avalie se a é menor ou igual a b	0
!=	a ! = b	Avalie se a é diferente de b	1

Link

Para mais detalhes da Álgebra Booleana, leia Scheinerman (2003), e o tipo bool em C. Acesse:

https://goo.gl/3gSrrh

Por fim, temos os operadores lógicos que nos proporcionam escrever expressões lógicas, as quais realizam funções lógicas, dentre as quais temos: e, ou e não. Além de um operador lógico, também conhecido como conectivo, uma expressão lógica é formada por duas proposições.

Uma proposição é uma afirmação que pode ser verdadeira ou falsa. Assim, uma expressão lógica é formada pela conexão de duas ou mais proposições por meio de um conector. Existe uma maneira de mapear todas as possibilidades de valores dessas proposições, bem como o valor da expressão — essa forma de mapeamento é denominada tabela-verdade.

Cada operador tem uma tabela-verdade. A seguir, você pode ver esse mapeamento com os 3 operadores, bem como os símbolos que os representam: e (&&), ou (||) e não (!). Vamos nomear uma das proposições de P e a outra de T, e, neste primeiro momento, ter true(1) ou false(0) como valores dessas proposições. Observe as Tabelas 3, 4 e 5.

Tabela 3. Tabela-verdade da operação E Lógico.

P	T	P && T
True	True	True
True	False	False
False	True	False
False	False	False

Tabela 4. Tabela-verdade da operação Ou Lógico.

P	T	P || T
True	True	True
True	False	True
False	True	True
False	False	False

Tabela 5. Tabela-verdade da operação Não Lógico.

P	! P
True	False
Fale	True

Da mesma forma do caso das operações e expressões relacionais, vamos usar bastante as tabelas-verdade e operadores lógicos quando estivermos tratando das estruturas de seleção.

Como mencionado anteriormente, existe um tipo de variável em C que representa o tipo booleano. Veja alguns detalhes, a seguir.

Para utilizá-la, é preciso a adição da biblioteca stdbool com a seguinte linha de código: # **include <stdbool.h>**

A declaração das variáveis é feita da seguinte forma: **bool a**.

A inicialização e atribuição podem ser feitas usando-se **true/false** ou **1/0.**

Atribuir o valor verdadeiro para a variável a: **a=true ; (a=1).**

A resposta das operações lógicas será SEMPRE **0** e **1.**

Para exemplificar, nas Figuras 8 e 9, segue um código-fonte e a execução de um programa que mostra o uso dos operadores lógicos: E, Ou e Não.

```c
1  #include <stdio.h>
2  #include <stdbool.h>
3
4  int main(void)
5  {
6      bool p=false, t=true;
7      printf("a: %d b: %d\n",p,t);
8      printf("a && b: %d\n", p&&t);
9      printf("a || b: %d\n", p||t);
10     printf("!a: %d\n", !p);
11 }
```

```
a: 0 b: 1
a && b: 0
a || b: 1
!a: 1
```

Figuras 8 e 9. Operadores lógicos.

Para finalizar, observe a Tabela 6, com operadores lógicos em C.

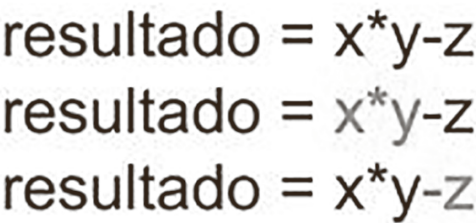
Tabela 6. Operadores lógicos.

Operação	Operador
E	&&
Ou	||
Não	!

Assim podemos fazer uma tabela-resumo (Tabela 7) com todos os operadores que foram vistos até agora:

Tabela 7. Resumo de todos os operadores: aritméticos, relacionais e lógicos.

Tipo	Operação	Operador
Aritmético	Multiplicação	*
Aritmético	Divisão	/
Aritmético	Soma	+
Aritmético	Subtração	-
Aritmético	Módulo	%
Relacional	Menor que	<
Relacional	Menor que ou igual a	<=
Relacional	Maior que	>
Relacional	Maior que ou igual a	>=
Relacional	Igual a	==
Relacional	Diferente de	!=
Lógico	E	&&
Lógico	Ou	||
Lógico	Não	!

Expressões com vários operadores e precedência

As expressões aritméticas, relacionais e lógicas podem ter vários operadores, e, para isso, existe uma ordem de precedência entre os operadores do mesmo tipo. A primeira regra e mais importante é: se existir, em uma expressão, uma subexpressão dentro de um parênteses, essa expressão deve ser a primeira a ser resolvida, ou seja, os parênteses, quando existentes, são sempre os primeiros a serem resolvidos.

Antes de avaliarmos uma expressão com vários tipos de operadores, vamos avaliar expressões com vários operadores, porém do mesmo tipo. Para isso, vamos usar a ordem de precedência respeitada pelo compilador da linguagem C. Entre esses operadores, começamos com os aritméticos. Observe a Tabela 8.

Tabela 8. Ordem de precedência dos operadores aritméticos.

Ordem de execução	Operação	Operador
1	Multiplicação	*
2	Divisão	/
3	Soma	+
4	Subtração	-
5	Módulo	%

O que isso quer dizer? Isso significa que, se em uma expressão existir uma multiplicação, uma divisão e uma soma, a ordem deve ser a seguinte: multiplicação, divisão e soma. Veja um exemplo na Figura 10.

```
resultado = x*y-z
resultado = x*y-z
resultado = x*y-z
```

Figura 10. Expressão com operadores aritméticos.

Nessa expressão, a multiplicação deve ser feita antes da subtração — basta seguir a regra de precedência da tabela mostrada anteriormente. Da mesma forma, existe uma ordem de precedência respeitada pelo compilador C, entre os operadores relacionais. Essa ordem pode ser observada na Tabela 9, a seguir.

Tabela 9. Ordem de precedência dos operadores relacionais.

Ordem de execução	Operação	Operador
1	Menor que	<
2	Menor que ou igual a	<=
3	Maior que	>
4	Maior que ou igual a	>=
5	Igual a	==
6	Diferente de	!=

Vamos a um exemplo. Observe a Figura 11.

```
a > b == c < d
a > b == c < d
a > b == c < d
a > b == c < d
```

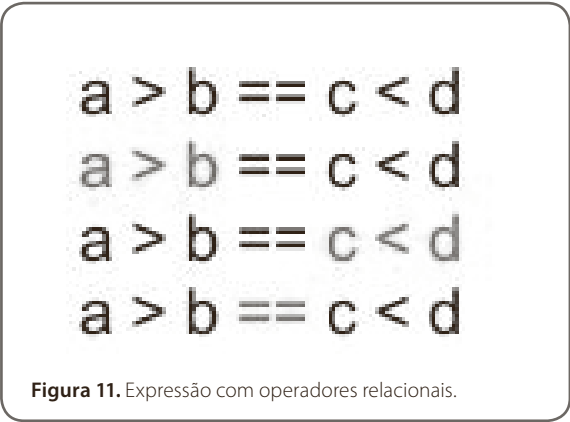



Figura 11. Expressão com operadores relacionais.

Nessa expressão, as operações relacionais de maior e menor de cada lado da operação igual a devem ser feitas primeiro. Logo, em seguida, comparam-se os valores obtidos entre essas operações e, dependendo do valor de a, b, c e d, teremos um valor verdadeiro (true,1) ou falso (false,0).

Para finalizar, temos a ordem de precedência dos nossos operadores lógicos. Observe a Tabela 10.

Tabela 10. Ordem de precedência dos operadores lógicos.

Ordem de execução	Operação	Operador
1	E	&&
2	Ou	||
3	Não	!

Para exemplificar, vamos usar a seguinte expressão, conforme a Figura 12.

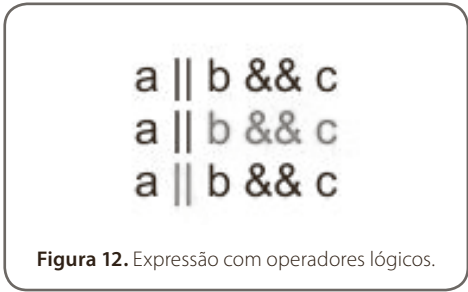



Figura 12. Expressão com operadores lógicos.

Seguindo as regras, a operação E é feita antes, para só em seguida a operação Ou ser executada. Agora, vamos colocar todos os operadores — aritméticos, relacionais e lógicos — em uma única tabela e reorganizar a ordem de precedência. Veja na Tabela 11.

Tabela 11. Ordem de precedência de todos os operadores.

Ordem de execução	Operação	Operador
1	Multiplicação	*
2	Divisão	/
3	Soma	+
4	Subtração	-
5	Módulo	%
6	Menor que	<
7	Menor que ou igual a	<=
8	Maior que	>
9	Maior que ou igual a	>=
10	Igual a	==
11	Diferente de	!=
12	E	&&
13	Ou	||
14	Não	!

A ordem de precedência é a seguinte:

- As operações aritméticas são realizadas primeiramente.
- Em seguida, são realizadas as operações relacionais.
- O próximo passo é realizar operações lógicas, na seguinte ordem (ver Figura 13).

Exemplificamos, descobrindo o valor da expressão: (y + z < x) OU (x>15) E (y < 55), sendo x igual a 15, y igual a 70 e z igual a 5.

```
(y + z < x) || (x>15) && (y < 55)
(y + z < x) || (x>15) && (y < 5)
(y + z < x) || (x>15) && (y < 55)
(y + z < x) || (x>15) && (y < 55)
(y + z < x) || (x>15) && (y < 55)
(y + z < x) || (x>15) && (y < 55)
(y + z < x) || (x>15) && (y < 55)
```

Figura 13. Expressão com operadores aritméticos, relacionais e lógicos.

- Primeiro, precisamos avaliar todas as operações que estão em parênteses.
- O primeiro passo é realizar a operação y+z, que é igual a 75.
- Agora, é preciso avaliar as operações relacionais.
  - (y+z<x) : (75<15), que é falso.
  - (x>15): (15>15), que é falso.
  - (y<55): (70<55) que é falso.
- Se colocarmos os valores lógicos na expressão inicial, temos:
  - (falso) ou (falso) e (falso)
- A operação e é realizada primeiro e, pela tabela-verdade, é possível ver que, quando temos dois valores falsos, o resultado é falso.
- Assim, a expressão fica: falso ou falso, resultando em falso, e esse é o valor final dessa expressão.

Uma dica para evitar confusão é definir as precedências usando parênteses. As expressões dentro dos parênteses mais internos são resolvidas primeiramente.

Operadores de pré e pós-incremento/ decremento

Por fim e não menos importante, nós temos, neste capítulo, operadores de incremento e decremento. Estas operações são utilizadas para acelerar o processo de aumentar ou diminuir unidades de variáveis.

O operador de incremento ++ adiciona a 1 a variável que utiliza esse operador. Já o operador de decremento -- subtrai 1 da variável.

Essas operações podem ser feitas antes ou depois do uso da variável, ou seja, um pré ou pós-incremento e um pré ou pós-decremento. No pré-incremento (decremento), o valor será modificado na instrução que a variável está sendo avaliada. Já, no caso de pós, o valor da variável será modificado na próxima instrução.

Para exemplificarmos, temos o seguinte código e sua saída de execução. Veja as Figuras 14 e 15.

```
1  #include <stdio.h>
2
3  int main(){
4      int i, i2;
5      i=10;
6       /* incremento */
7      printf ("valor de i: %d\n",i);
8      printf ("valor de i: pré incremento: %d\n",++i);
9      printf ("valor de i: pós incremento: %d\n",i++);
10     printf ("valor de i: %d\n",i);
11       /* decremento */
12     i2=20;
13     printf ("valor de i2: %d\n",i2);
14     printf ("valor de i2: pré decremento: %d\n",--i2);
15     printf ("valor de i2: pós decremento: %d\n",i2--);
16     printf ("valor de i2: %d\n",i2);
17     return 0;
18 }
```

Figura 14. Código-fonte para exemplificar o uso dos operadores de incremento e decremento e suas formas de uso.

```
valor de i: 10
valor de i: pré incremento: 11
valor de i: pós incremento: 11
valor de i: 12
valor de i2: 20
valor de i2: pré decremento: 19
valor de i2: pós decremento: 19
valor de i2: 18
```

Figura 15. Saída da execução do código-fonte para exemplificar o uso dos operadores de incremento e decremento e suas formas de uso.

Uma das grandes utilidades dos operadores de incremento ou decremento é no uso de variáveis contadoras. Imagine que você tenha que desenvolver um sistema de fichas para uma central de atendimento. Os números de ficha entregues para os usuários nada mais são do que valores que, a cada solicitação, aumenta-se 1, ou seja, vamos contando de 1 em 1.

Vamos criar um programa que conte de 1 até 5 de duas formas: como a operação de soma, usando o operador +, e com a operação de incremento, usando o operador ++. Veja as duas formas nos códigos da Figura 16, a seguir.

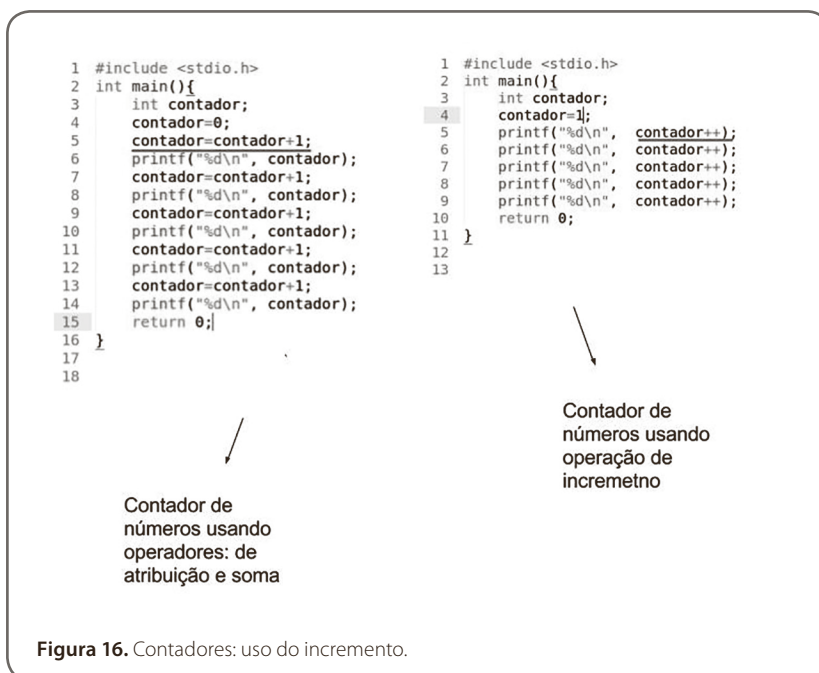



Figura 16. Contadores: uso do incremento.

Os dois programas realizam a mesma tarefa: contam de 1 a 5 e mostram esta contagem na tela. A diferença está na forma que isso pode ser feito. A forma reduzida é bastante utilizada em programação, e outro detalhe interessante é que podemos também incrementar valores diferentes de 1. Imagine contar de 1 a 10, somente os números pares. A seguir, na Figura 17, serão mostradas as duas formas de se realizar essa tarefa: forma convencional e forma reduzida.

```c
1  #include <stdio.h>
2  int main(){
3      int contador;
4      contador=0;
5      printf("%d\n", contador);
6      contador=contador+2;
7      printf("%d\n", contador);
8      contador=contador+2;
9      printf("%d\n", contador);
10     contador=contador+2;
11     printf("%d\n", contador);
12     contador=contador+2;
13     printf("%d\n", contador);
14     contador=contador+2;
15     printf("%d\n", contador);
16     return 0;
17 }
```

Contador de números pares, operação convencional

```c
1  #include <stdio.h>
2  int main(){
3      int contador;
4      contador=0;
5      printf("%d\n",  contador);
6      printf("%d\n",  contador+=2);
7      printf("%d\n",  contador+=2);
8      printf("%d\n",  contador+=2);
9      printf("%d\n",  contador+=2);
10     printf("%d\n",  contador+=2);
11     return 0;
12 }
```

Contador de números pares usando o incremento +=2

Figura 17. Contadores: uso do incremento, para um contador de números pares.

Referências

BOOLEAN type support library. *Cppreference.com, 2017.* Disponível em: <http://en.cppreference.com/w/c/types/boolean>. Acesso em: 22 fev. 2018.

SCHEINERMAN, E. R. *Matemática Discreta:* uma introdução. São Paulo: Thomson Pioneira, 2003.

Leituras recomendadas

PAES, R. B. *Introdução à Programação com a Linguagem C*. São Paulo: Novatec, 2016. 296p.

PINHEIRO, F. A. C. *Elementos de programação em C*. Porto Alegre: Bookman, 2012. 548p.

DICA DO PROFESSOR

Como usar operandos aritméticos e relacionais para resolver um problema real utilizando a linguagem C? Acompanhe o vídeo da Dica do Professor para receber orientações sobre o uso desses operadores e uma dica especial sobre o tipo booleando na linguagem C.

Conteúdo interativo disponível na plataforma de ensino!

EXERCÍCIOS

1) Considerando o trecho de código escrito na linguagem C, qual é o valor da variável resultado1?

```c
1  #include <stdio.h>
2  int main(){
3      int a=8;
4      float b=7.5;
5      int resultado1;
6      resultado1= a*a-b;
7      printf("%d\n", resultado1);
8      return 0;
9  }
```

A) 56.5.

B) É impossível determinar, pois em C não se pode fazer operações aritméticas com tipos diferentes.

C) 56.

D) 7.

E) 7.5.

2) **Considerando o trecho de código, qual é o valor que será impresso na tela ao final da execução?**

A) 7.

B) 7.5.

C) 7.00.

D) Não se pode realizar operações com números inteiros e armazenar em uma variável do

tipo float. Portanto, esse código não é executado, havendo um erro de sintaxe.

E) Não é possível determinar, pois a variável resultado não foi inicializada.

3) Avalie o seguinte código, e diga qual é o resultado que será impresso.

```c
1  #include <stdio.h>
2  int main(){
3      float n=3.2;
4      float i=2.5;
5      float resultado=10;
6      resultado=resultado+i+n*3-2;
7      printf("%.2lf\n",resultado);
8      return 0;
9  }
```

A) 20.10.

B) 11.

C) 8.

D) 15.7.

E) 10.10.

4) Assinale a opção que indica a diferença entre os operadores = e ==, ambos utilizados na linguagem C.

A) = é um operando de atribuição, e == é um operando relacional que avalia se duas variáveis são iguais.

B) Não existe diferença, os dois operandos são de atribuição.

C) Não existe diferença, os dois operandos são relacionais, utilizados para avaliar se dois valores são iguais.

D) == é um operando de atribuição, e = é um operando relacional que avalia se duas variáveis são iguais.

E) Esses operandos não existem na linguagem C.

5) Qual é a função dos operadores ++ e -- na linguagem C, respectivamente?

A) São operadores de incremento e decremento.

B) É o operador de soma e de subtração, respectivamente.

C) É o operador de decremento e de incremento, respectivamente.

D) Esses operadores não existem em C.

E) São operadores de multiplicação e de divisão, escritos de forma diferente.

NA PRÁTICA

Fernando foi contratado para desenvolver um programa para auxiliar o RH de uma empresa do segmento de vestuário com o cálculo de descontos e salários para uma folha de contra-cheque.

Após analisar a solicitação da empresa, Fernando concluiu que poderia utilizar operadores em programação C para resolver o problema.

Conteúdo interativo disponível na plataforma de ensino!

Com o programa desenvolvido por Fernando, a empresa conseguiu fazer a análise da folha de pagamento dos funcionários de forma satisfatória.

SAIBA MAIS

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Operadores aritméticos e relacionais

No vídeo, você vai ver o uso de operadores para desenvolver um programa no qual são usados operadores aritméticos e relacionais.

Conteúdo interativo disponível na plataforma de ensino!

Operadores lógicos e relacionais

No vídeo, você vai conhecer mais sobre operadores lógicos e relacionais.

Conteúdo interativo disponível na plataforma de ensino!

Programação em C

O que é o incremento de uma variável e para que ele serve? O vídeo vai mostrar o uso dos operadores de incremento e de decremento em C.

Conteúdo interativo disponível na plataforma de ensino!